COS20007
Object-Oriented Programming

Topic 06 Part A
Responsibility Driven Design

Learning Outcomes

- The importance of Responsibility Driven Design
- Understanding the three implementation steps in Responsibility Driven Design
- Applying Responsibility Driven Design in practice with a chess game example

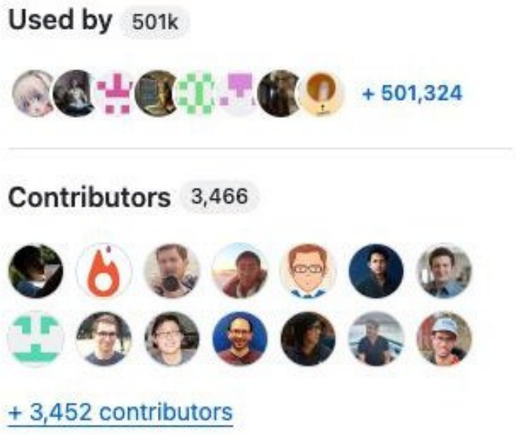
SWINBURNE UNIVERSITY OF TECHNOLOGY

Responsibility Driven Design

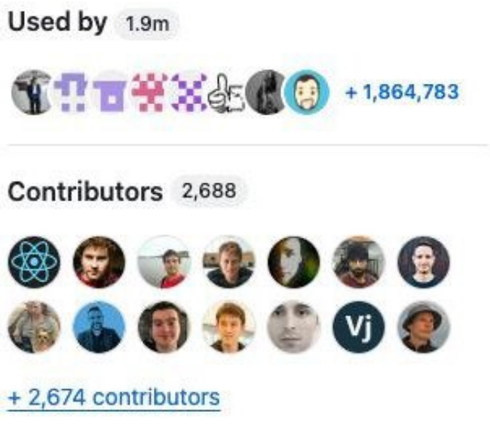
- Software development involves providing instructions for an unintelligent computer

- Developers work in teams to build software solutions, which typically contain millions of instructions

[Pytorch]



[Facebook React]



SWINBURNE UNIVERSITY OF TECHNOLOGY

Responsibility Driven Design

- Seeing how a solution will work requires clear communication
- Software design documentation before implementation plan
- Effective software design includes picturing the solution and having a common understanding

SWINBURNE UNIVERSITY OF TECHNOLOGY

Responsibility Driven Design

- How can developers picture the software solution and have a common understanding what the software does?
- How can software designers focus on standard communication protocols within the software, independently of implementation?
- How can we minimize the rework required for major design changes?
- How can we maximize encapsulation when defining blueprint classes?

SWINBURNE UNIVERSITY OF TECHNOLOGY

Responsibility Driven Design

- Creates effective OO designs using Roles, Responsibilities, and Collaborations
- Design before code = better code + faster written
- Emphasizes behavioral modeling
- Turns software requirements in to OO software

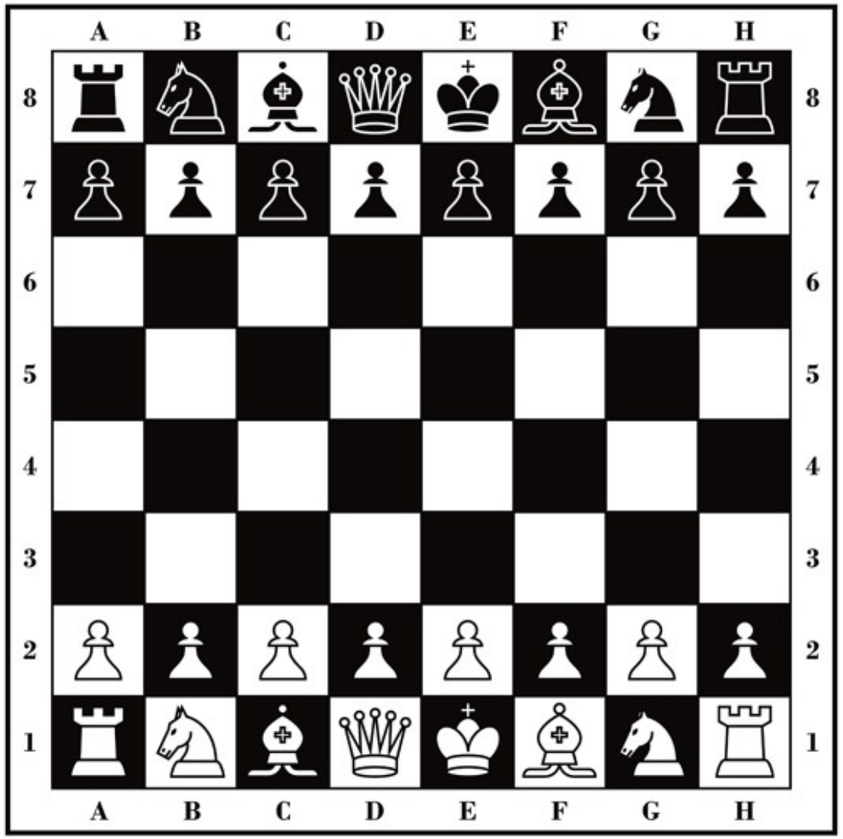
Wirfs-Brock, Rebecca et al. “Object Design: Roles, Responsibilities, and Collaborations.” (2002).

SWINBURNE UNIVERSITY OF TECHNOLOGY

Step 1: Define the purpose for objects in your program using **Roles**

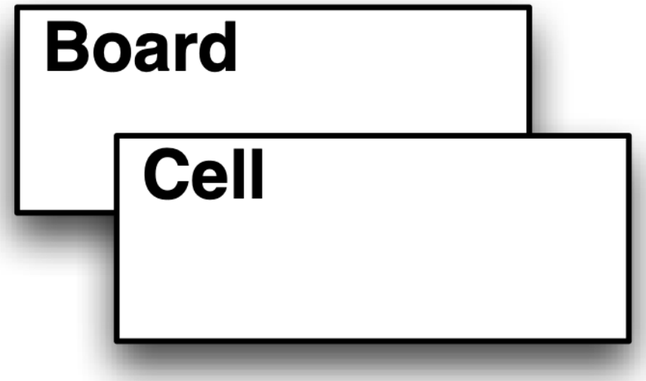
SWINBURNE UNIVERSITY OF TECHNOLOGY

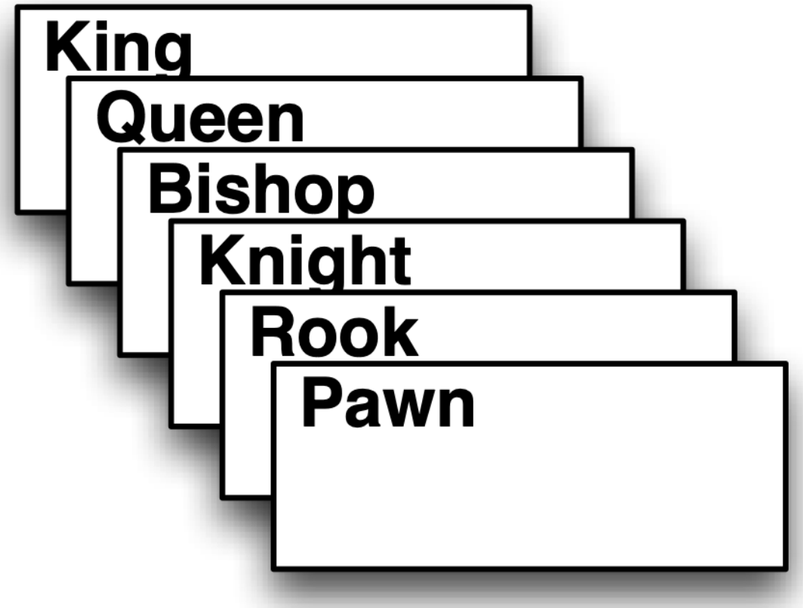
Picture the problem domain and identify **candidate** roles (nouns are a good start)



SWINBURNE UNIVERSITY OF TECHNOLOGY

Explore candidate roles using CRC cards

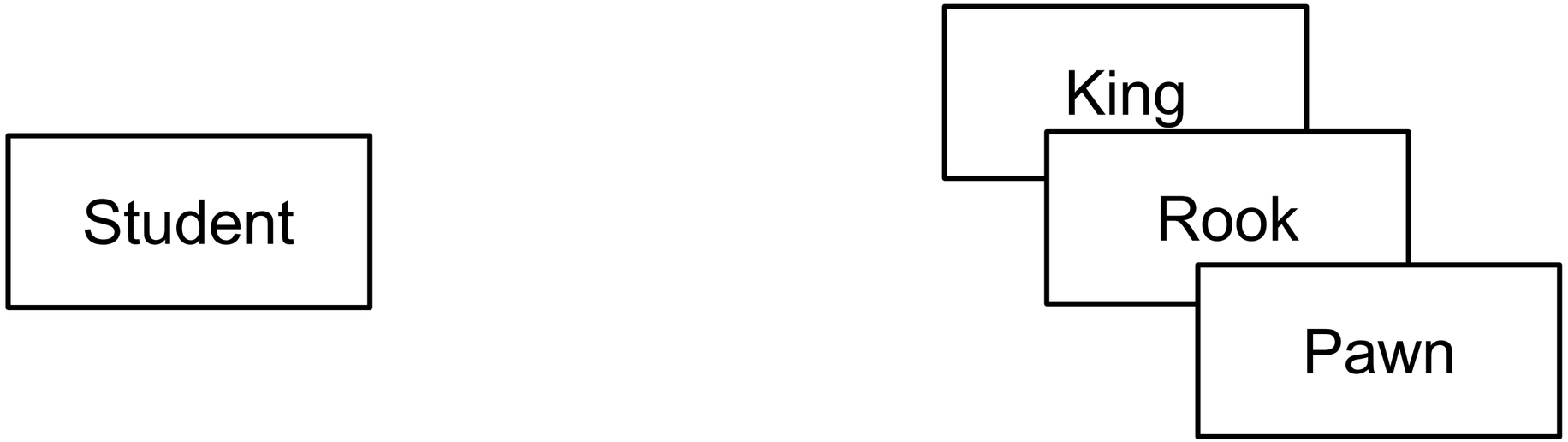




CRC = candidate role, responsibility, collaborations

SWINBURNE UNIVERSITY OF TECHNOLOGY

Draw boxes for classes in UML class diagrams to communicate static structure



SWINBURNE UNIVERSITY OF TECHNOLOGY

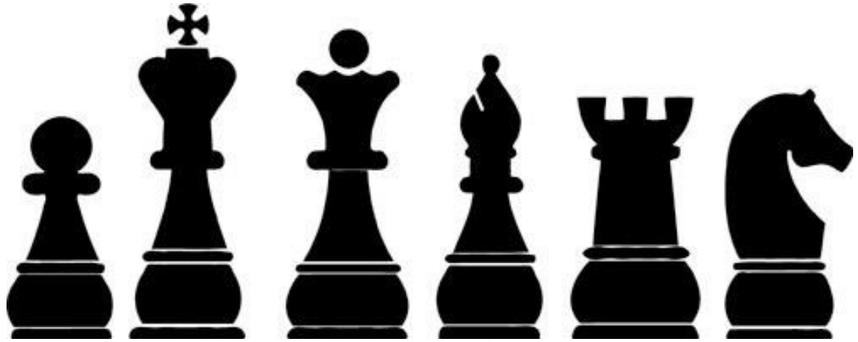
Step 2: Define **responsibilities** for each candidate role

- This step often causes you to jump back to the **Step 1**.

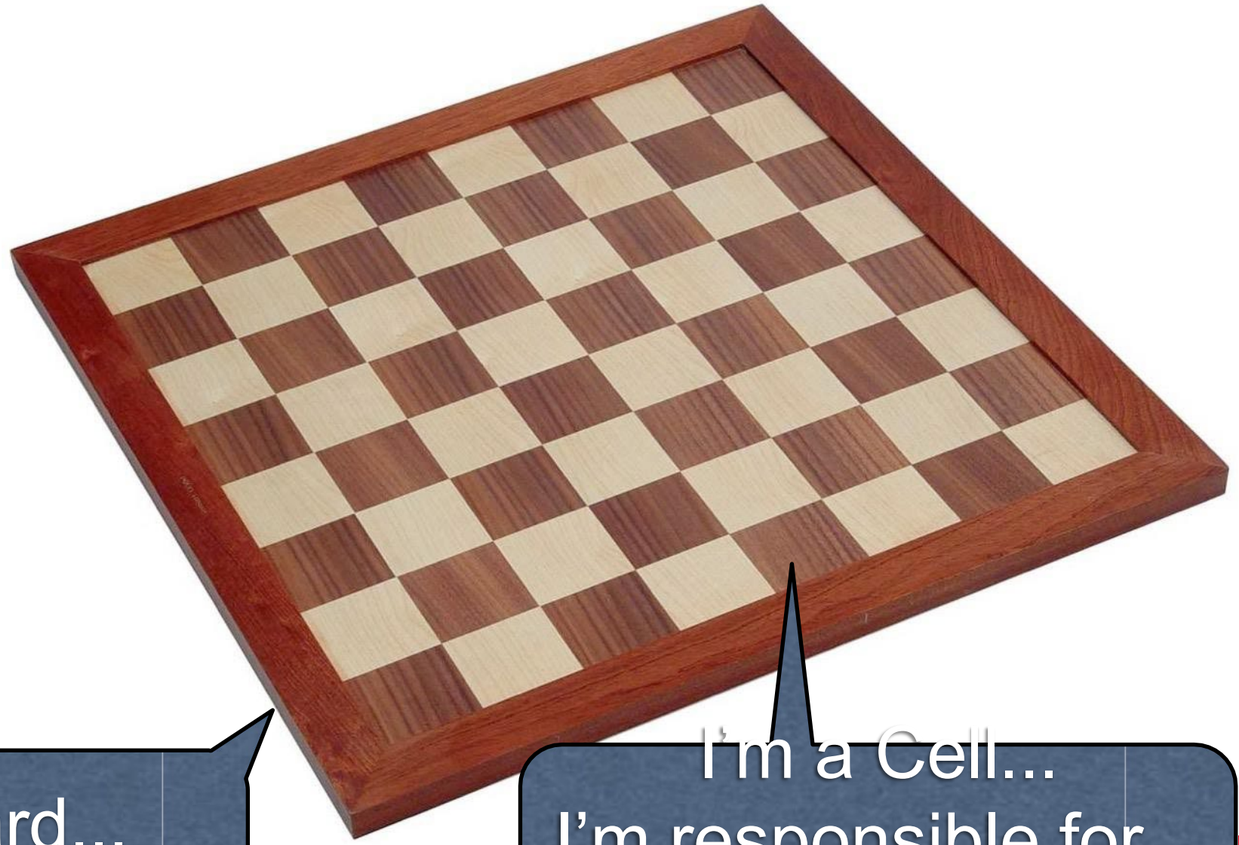
- Rethink what roles we really need
- Rethink what roles we do not need anymore
- Some roles may have many (similar) responsibilities

SWINBURNE UNIVERSITY OF TECHNOLOGY

Picture roles as having responsibilities within your overall solution



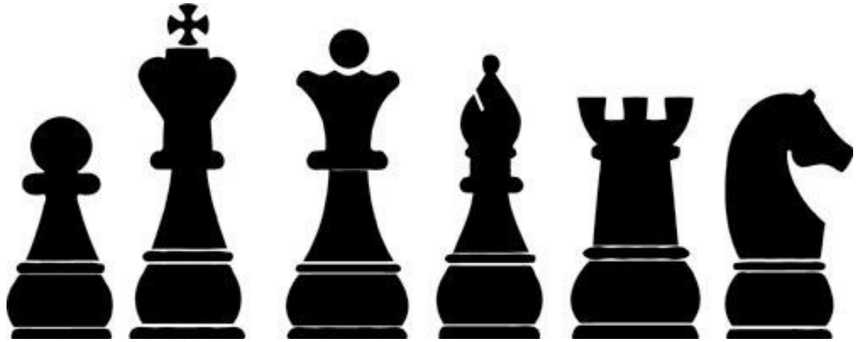
I'm a Bishop...
I'm responsible for...



I'm a Board...
I'm responsible for...

I'm a Cell...
I'm responsible for...

Include responsibilities to **know** things, this forms the data for your program

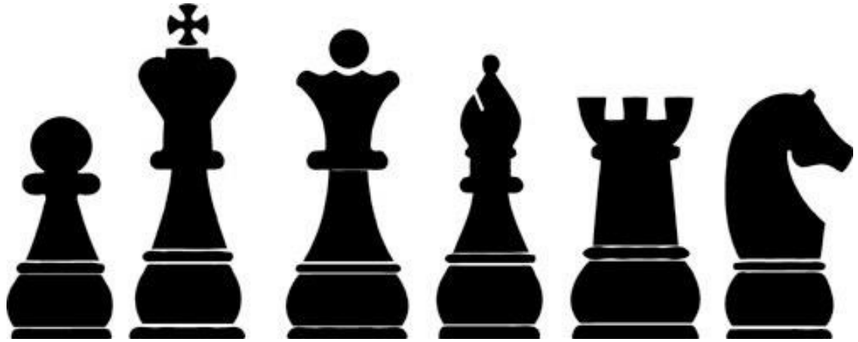


I'm a King...
I know my colour...

I'm a Board...
I know all of the cells.

I'm a Cell...
I know my occupant.

Include responsibilities to **do** things, these become methods in the solution



I'm a Pawn...
I can be a Queen.

I'm a Board...
I can move pieces.

I'm a Cell...
I can hold a piece.

Explore responsibilities using CRC cards

Pawn

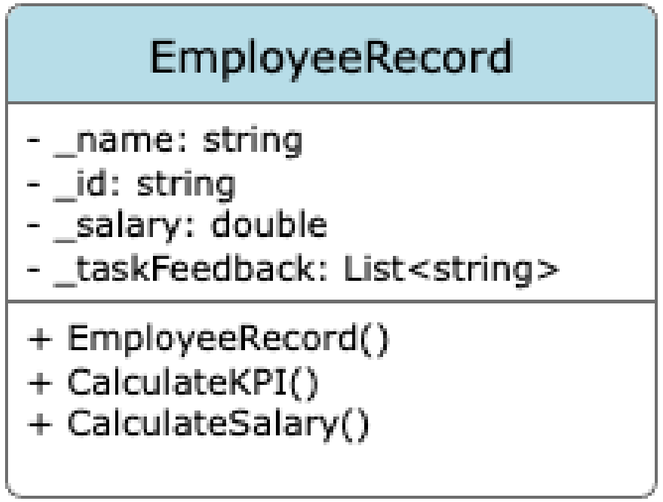
knows its color
knows its valid moves
can become a Queen
can take another piece

SWINBURNE UNIVERSITY OF TECHNOLOGY

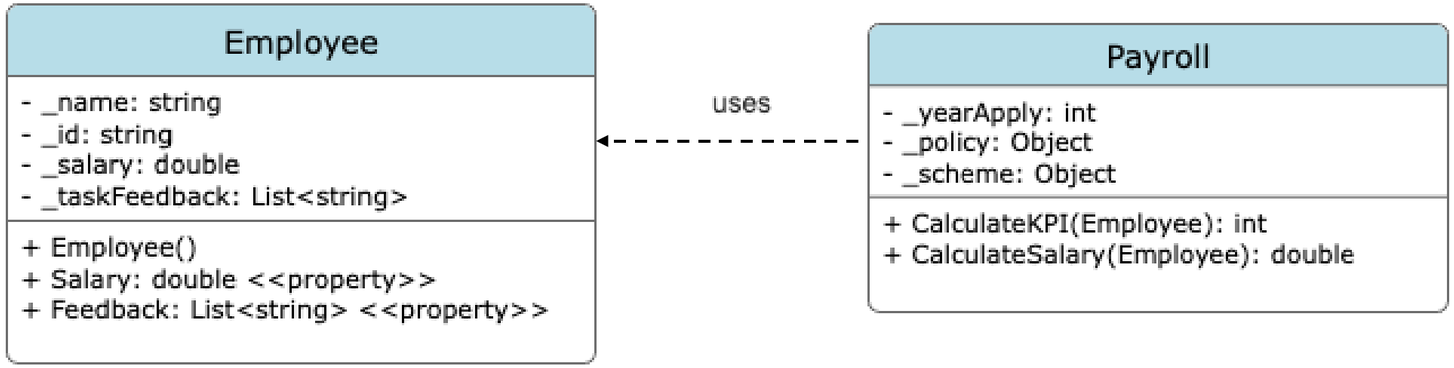
Step 2: Define **responsibilities** for each candidate role

Tips to clarify which roles we really need in our software.

Cohesion: the degree of strongly related functionality are described within the classes



Low Cohesion



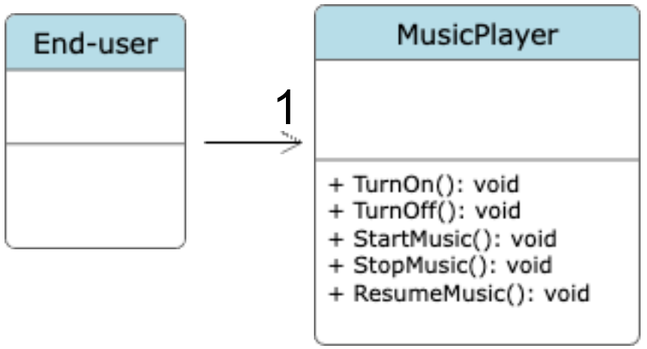
High Cohesion

SWINBURNE UNIVERSITY OF TECHNOLOGY

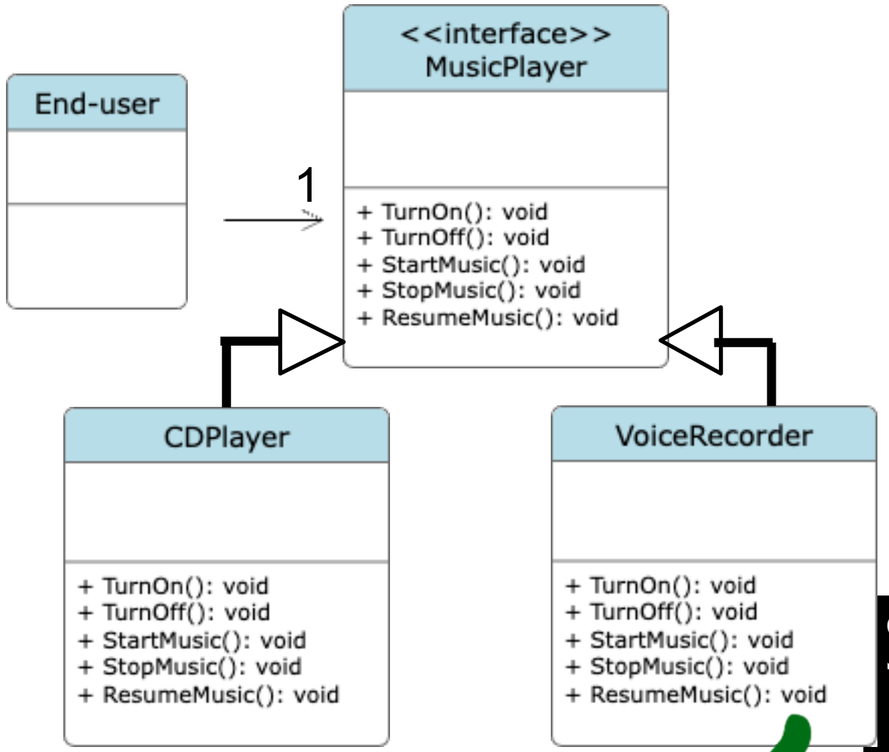
Step 2: Define **responsibilities** for each candidate role

Tips to clarify which roles we really need in our software.

Coupling: the degree of dependence among classes.



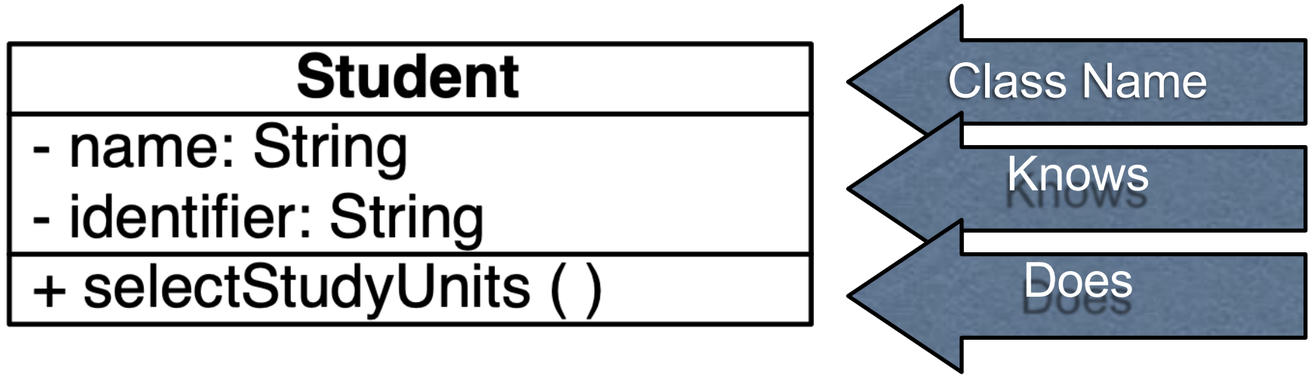
Tight Coupling

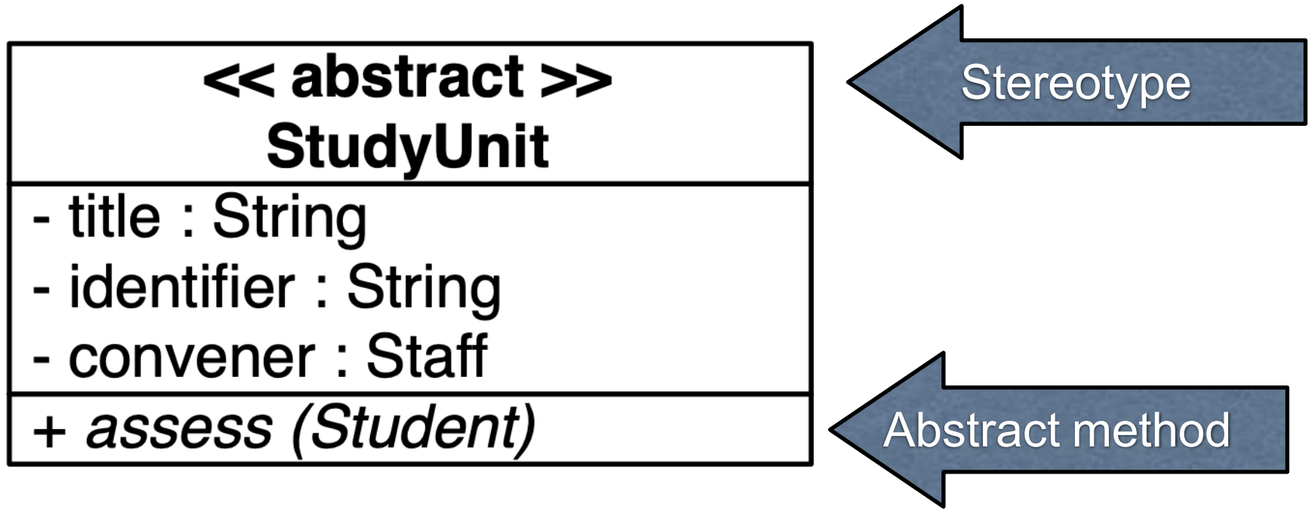


Loose Coupling

SWINBURNE UNIVERSITY OF TECHNOLOGY

Document responsibilities in UML class diagrams



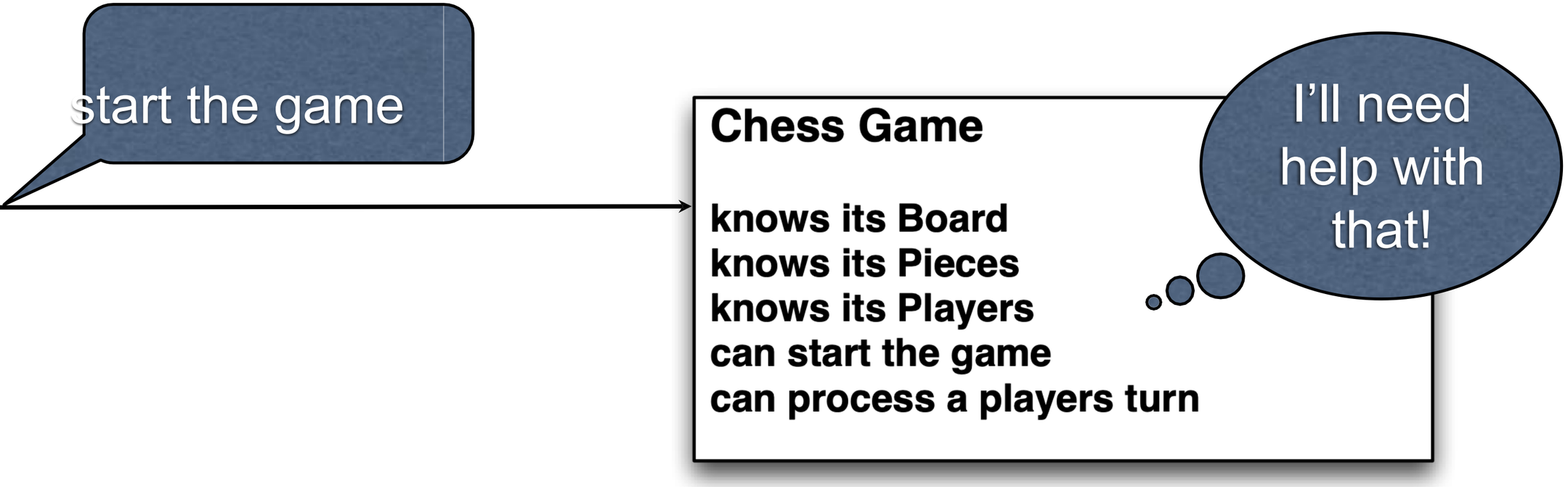


SWINBURNE UNIVERSITY OF TECHNOLOGY

Step 3: **Collaborate** with other objects to meet responsibilities

SWINBURNE UNIVERSITY OF TECHNOLOGY

When asked to perform a task, objects can ask others for help



SWINBURNE UNIVERSITY OF TECHNOLOGY

Think of collaborations as a **client/supplier** interaction or as a contract

Use the different kinds of relationships to help identify possible links

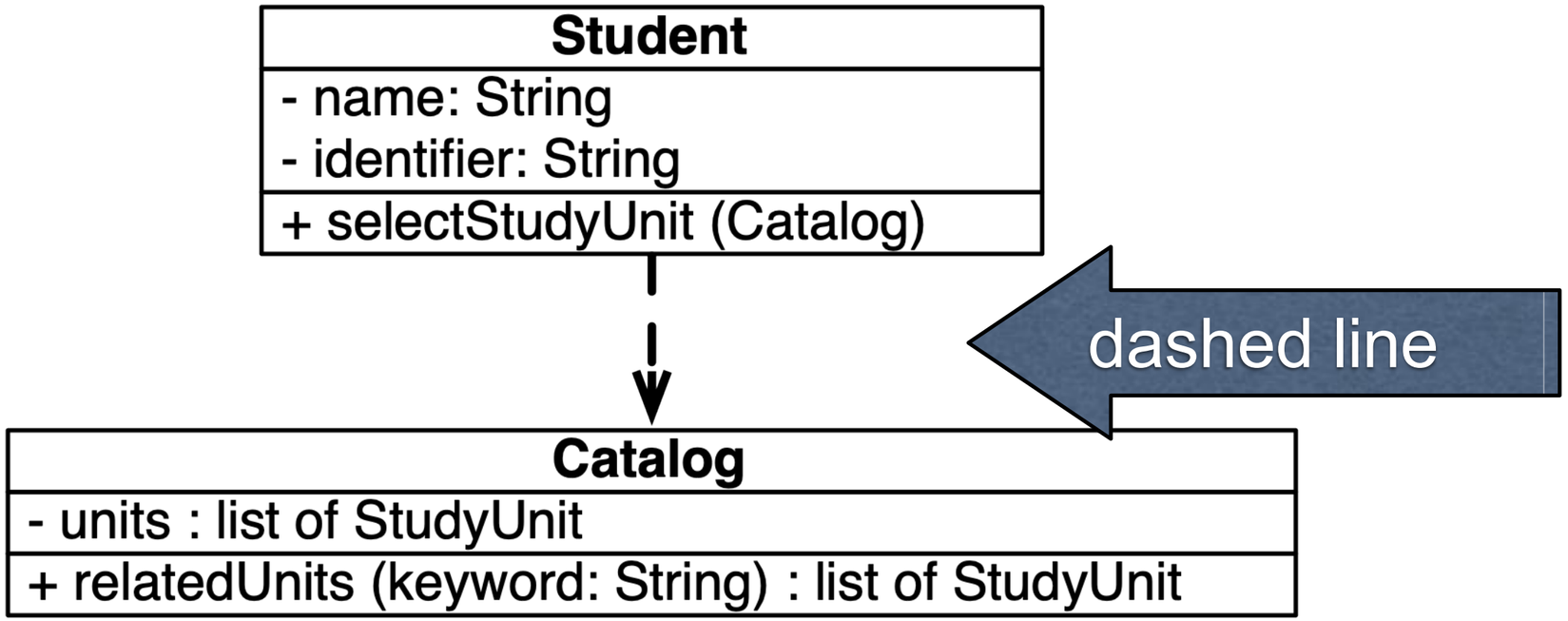
- Association
- Aggregation
- Dependency

→

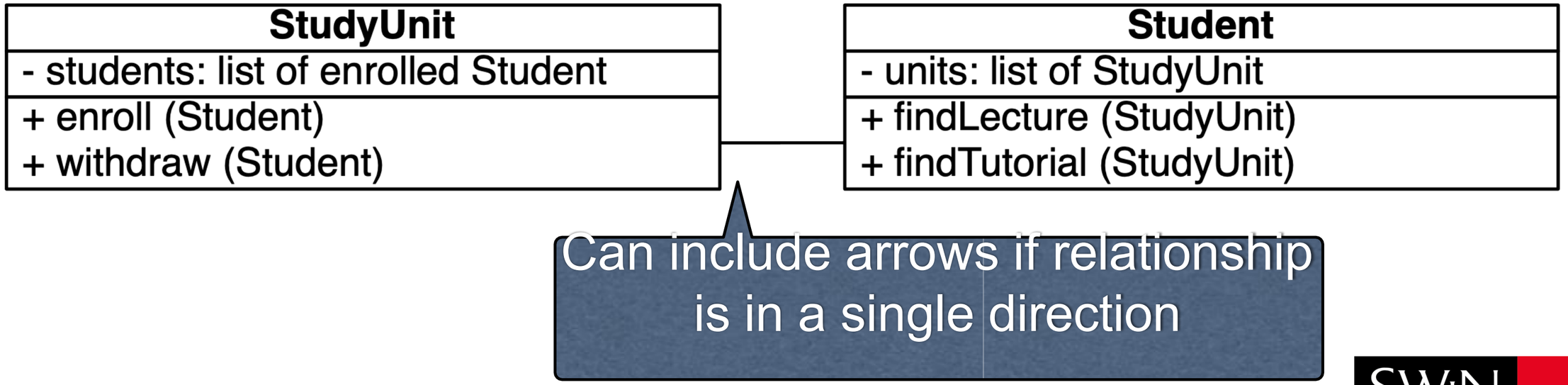
- where to use
- clarify using software requirement/business analysis

SWINBURNE UNIVERSITY OF TECHNOLOGY

Dependence involves temporary use of another object

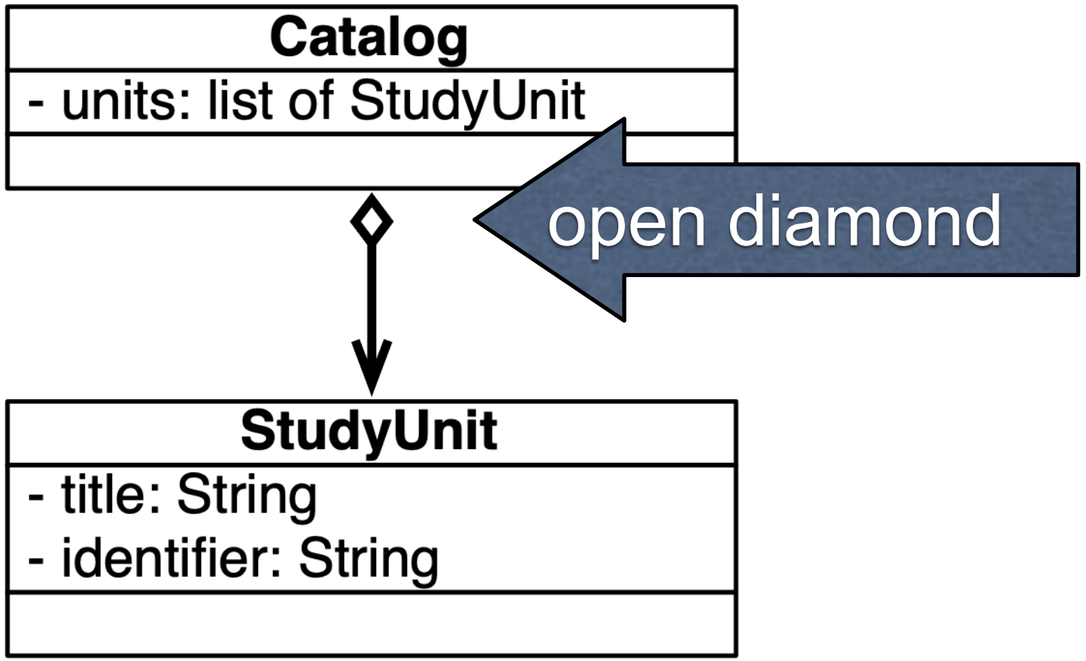


Permanent relationships are modelled as association using a solid line in UML



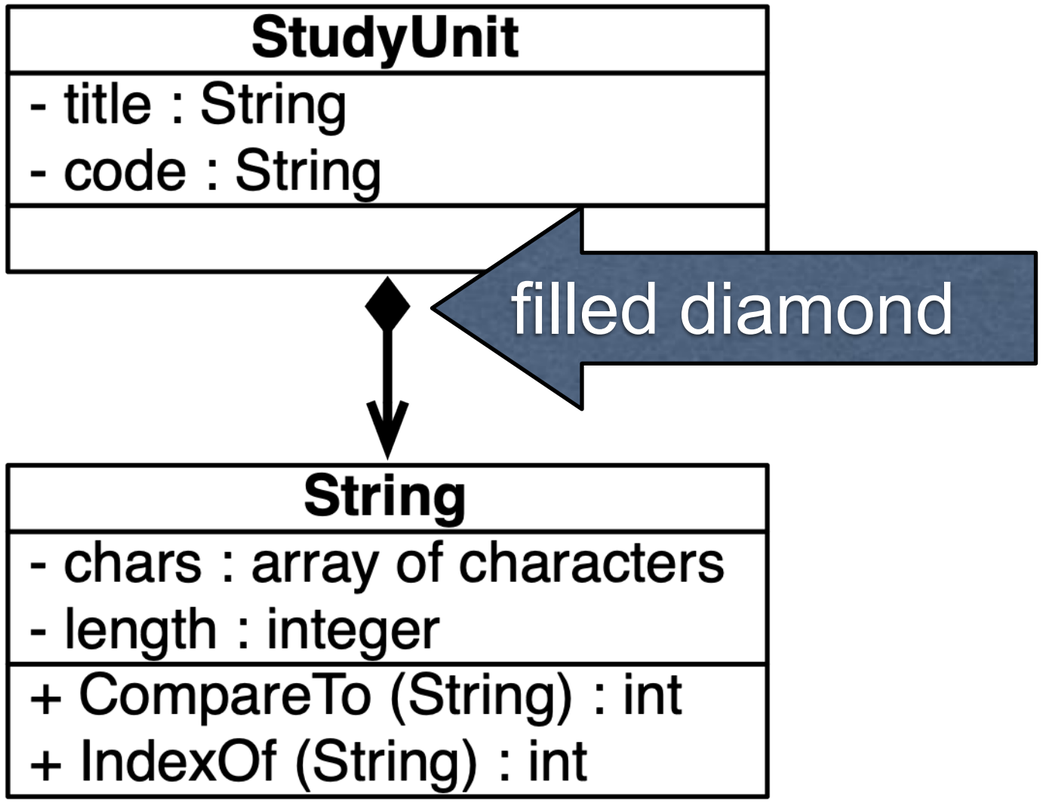
SWINBURNE UNIVERSITY OF TECHNOLOGY

Aggregation extends association to indicate a whole-part relation



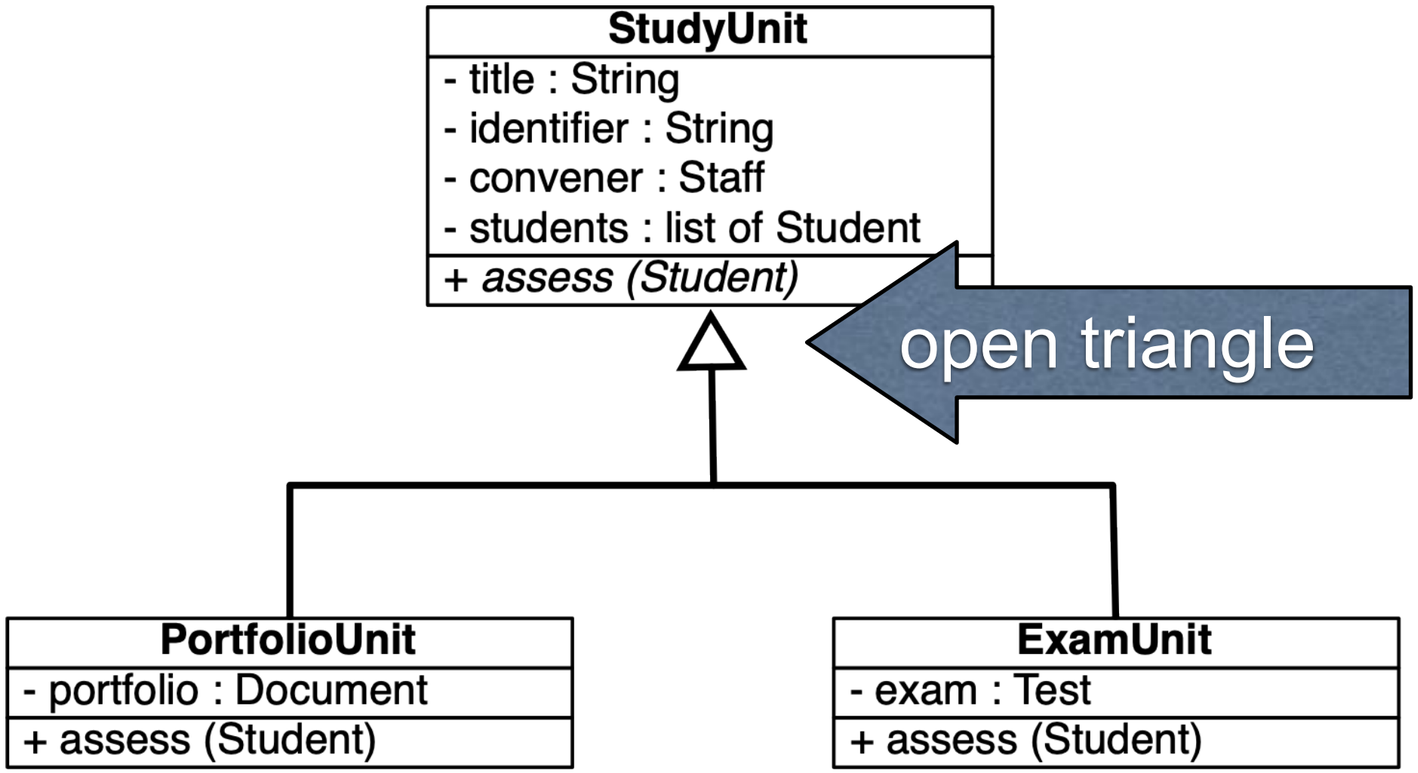
SWINBURNE UNIVERSITY OF TECHNOLOGY

Composition is a kind of aggregation, indicating destruction of the whole involves destruction of the part



SWINBURNE UNIVERSITY OF TECHNOLOGY

Inheritance captures class and interface inheritance for specialisation/generalisation



SWINBURNE UNIVERSITY OF TECHNOLOGY

Use scenarios to test how your model responds to events and implements features

Chess Game

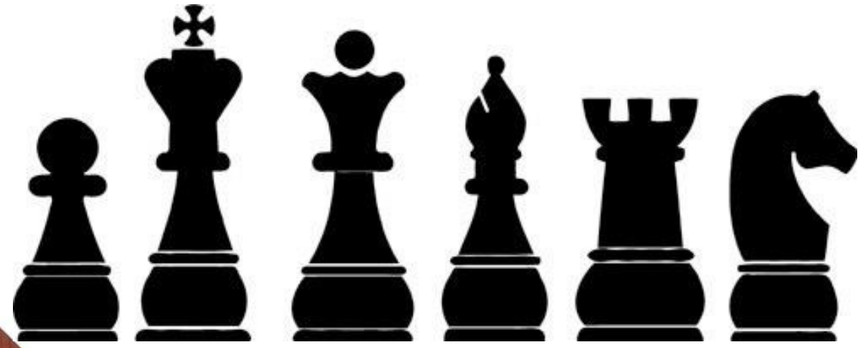
knows its Board
knows its Pieces
knows its Players
can start the game
can process a players turn

Board, setup.

Rook, exist.

Rook, this is your King.

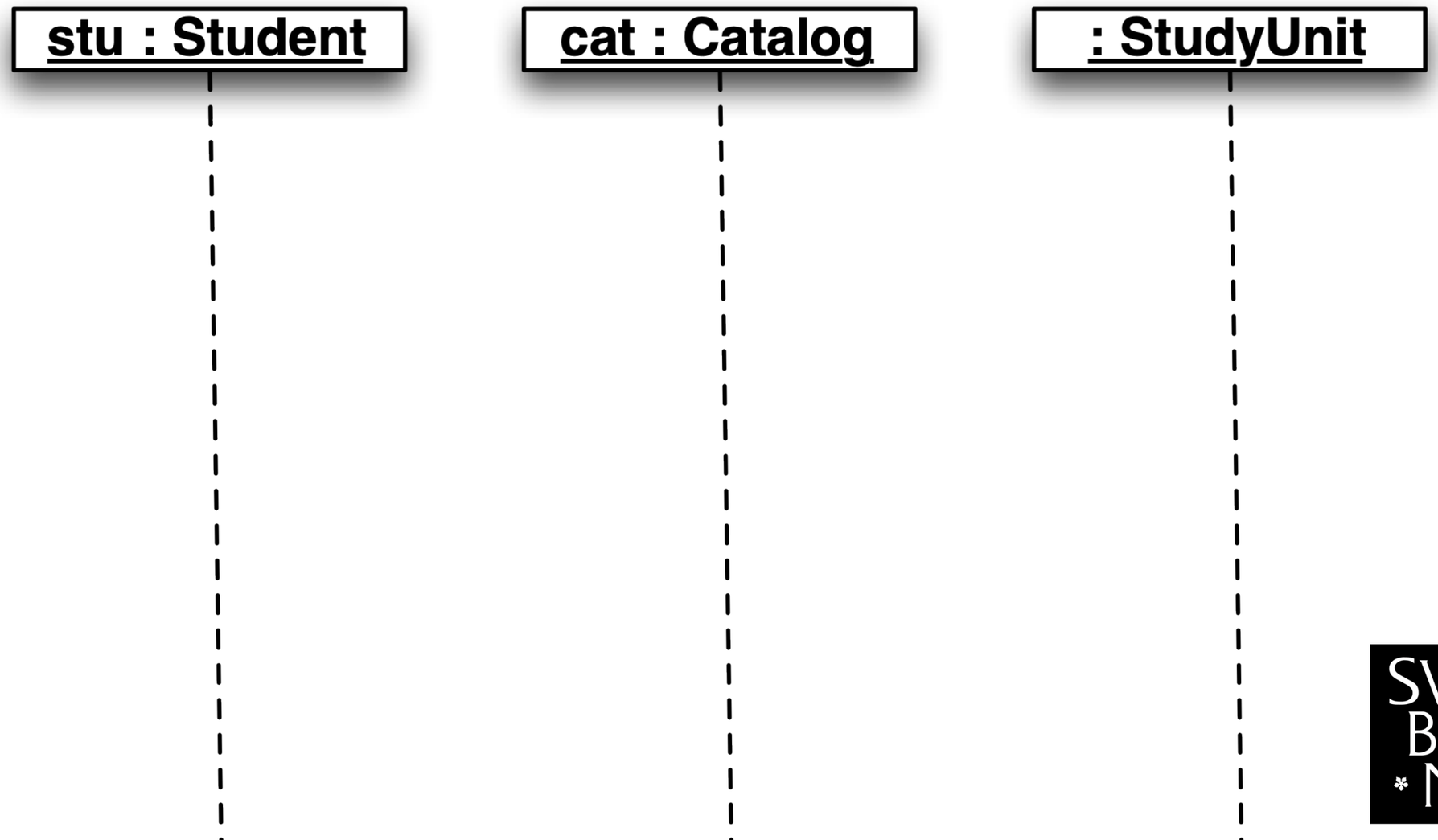
Cell, hold this Rook.



SWINBURNE UNIVERSITY OF TECHNOLOGY

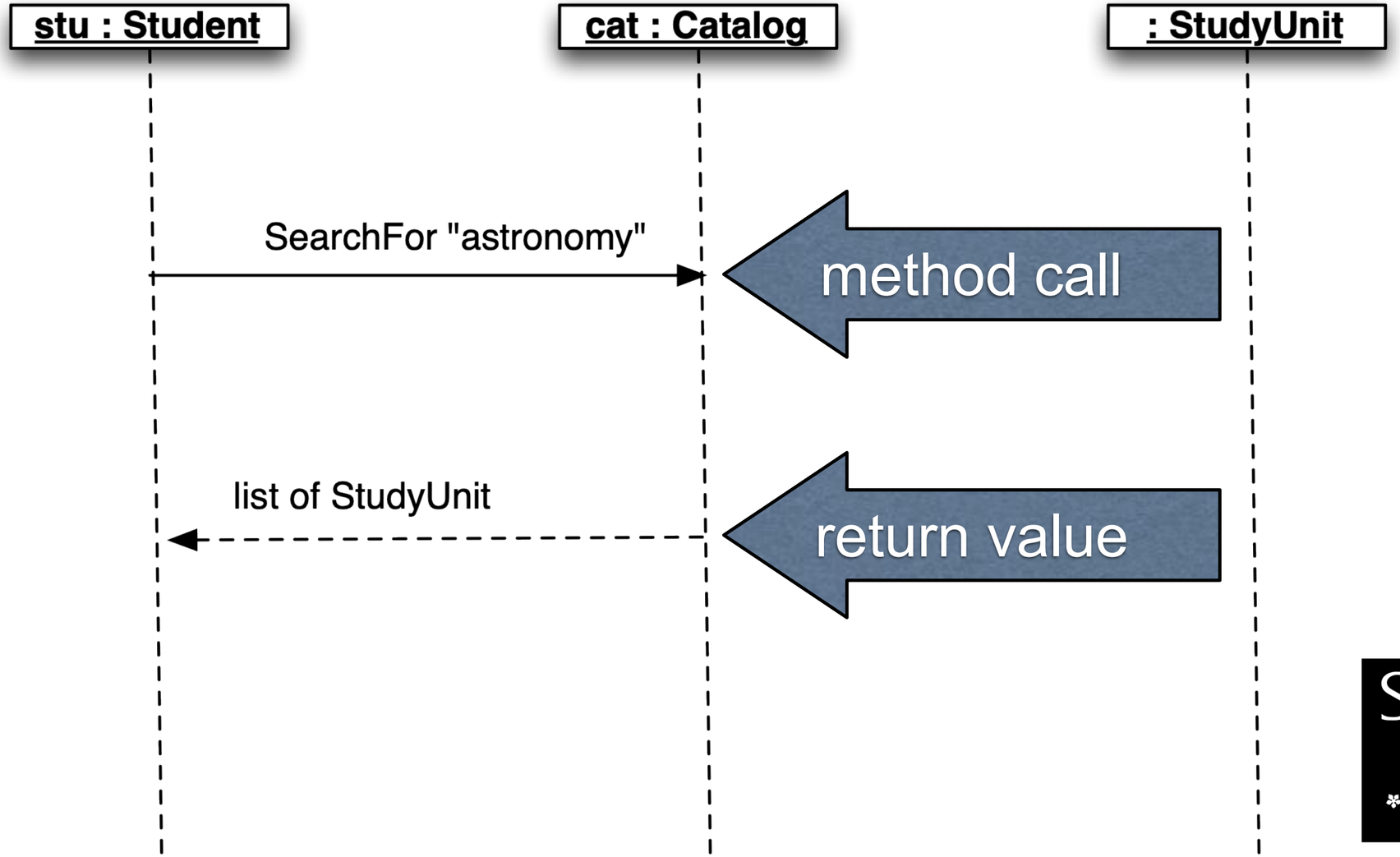
Communicate these dynamic interactions using sequence diagrams

Think of sequence diagrams as scripts, with life lines defining the existence of objects



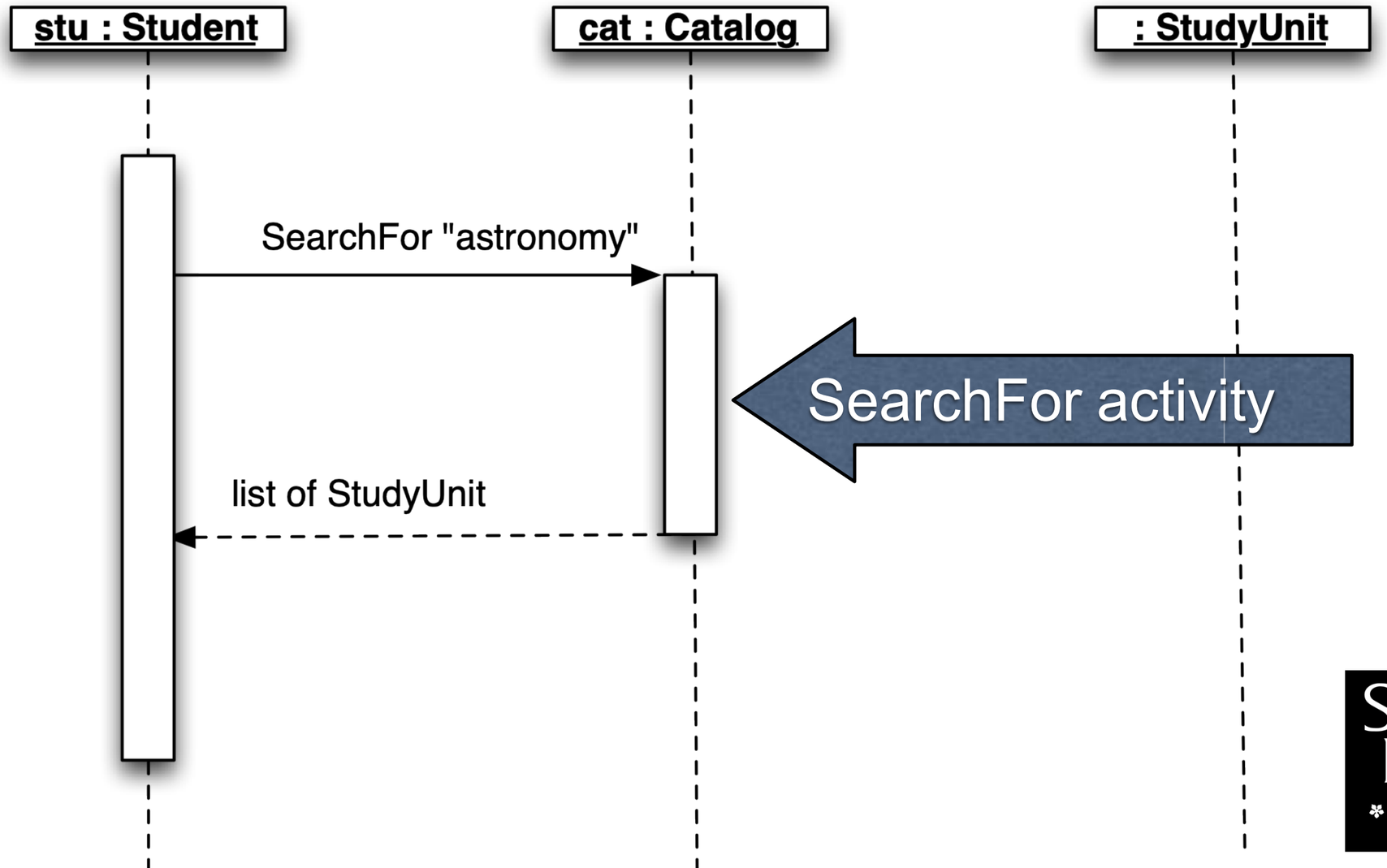
SWINBURNE UNIVERSITY OF TECHNOLOGY

Draw arrows between lifelines to show message passing (method calls)



SWINBURNE UNIVERSITY OF TECHNOLOGY

Use boxes to represent **activity**: when it is doing something or waiting for something to be done

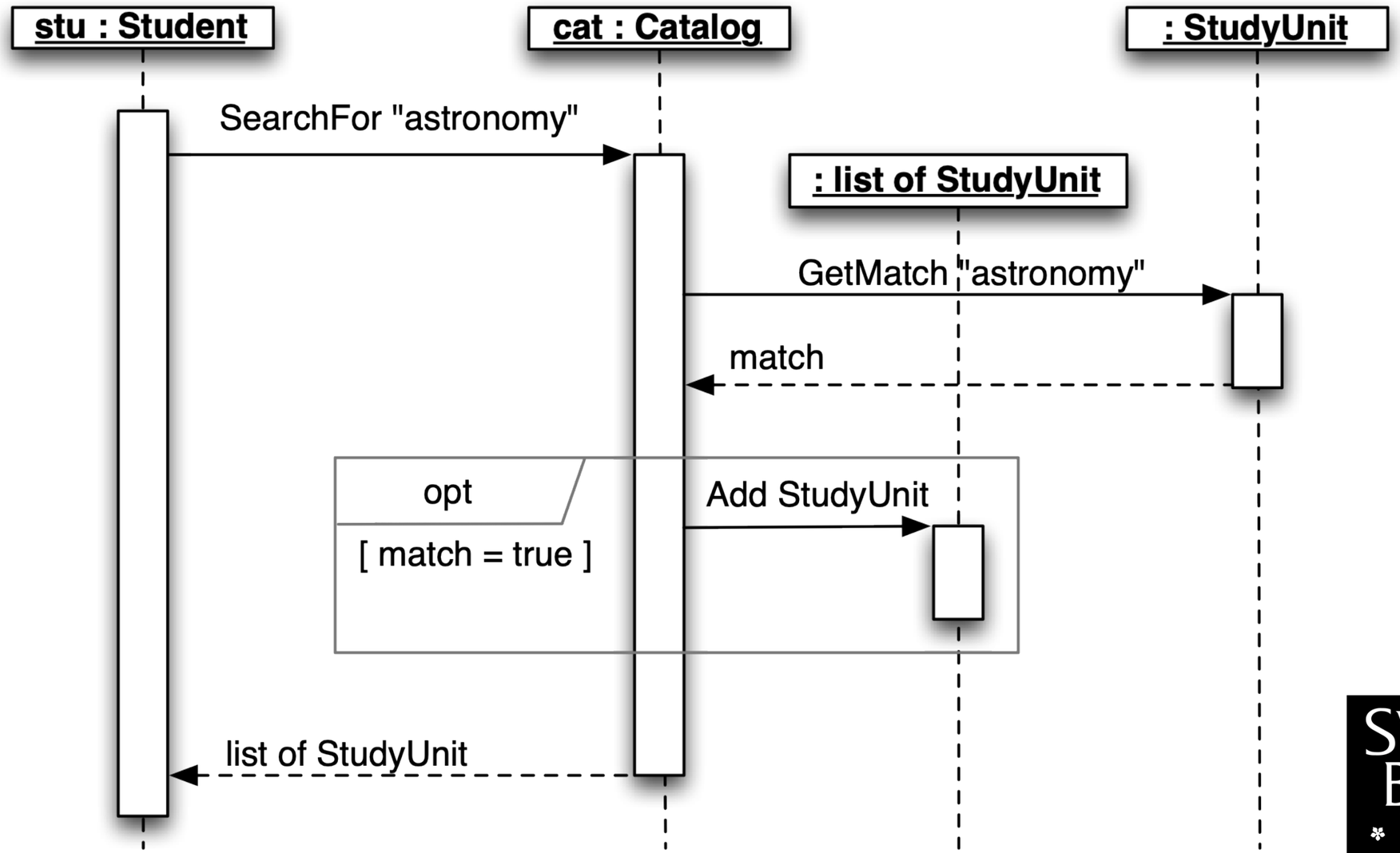


SWINBURNE UNIVERSITY OF TECHNOLOGY

Show control flow logic using **combination fragments**

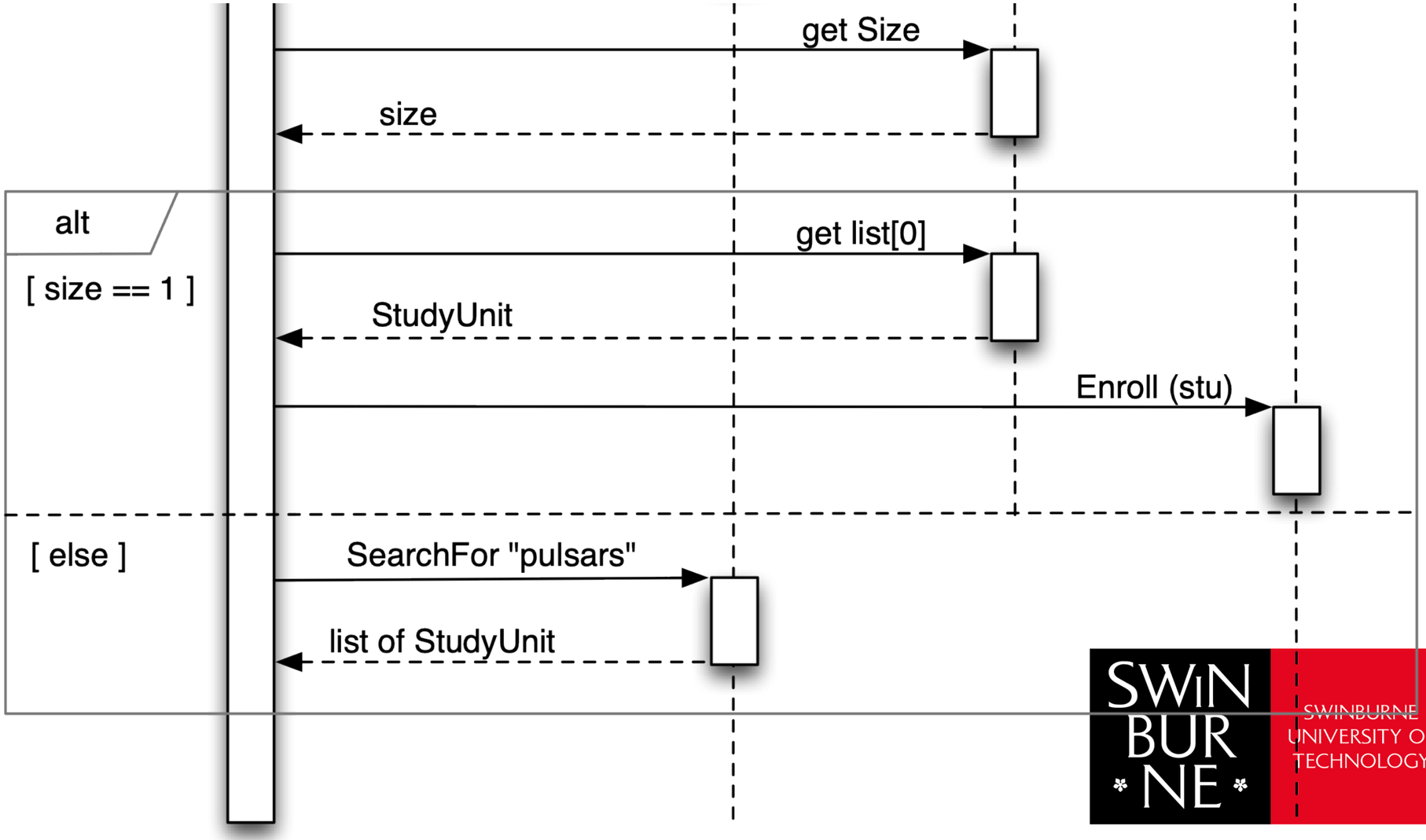
SWINBURNE UNIVERSITY OF TECHNOLOGY

Model if using options

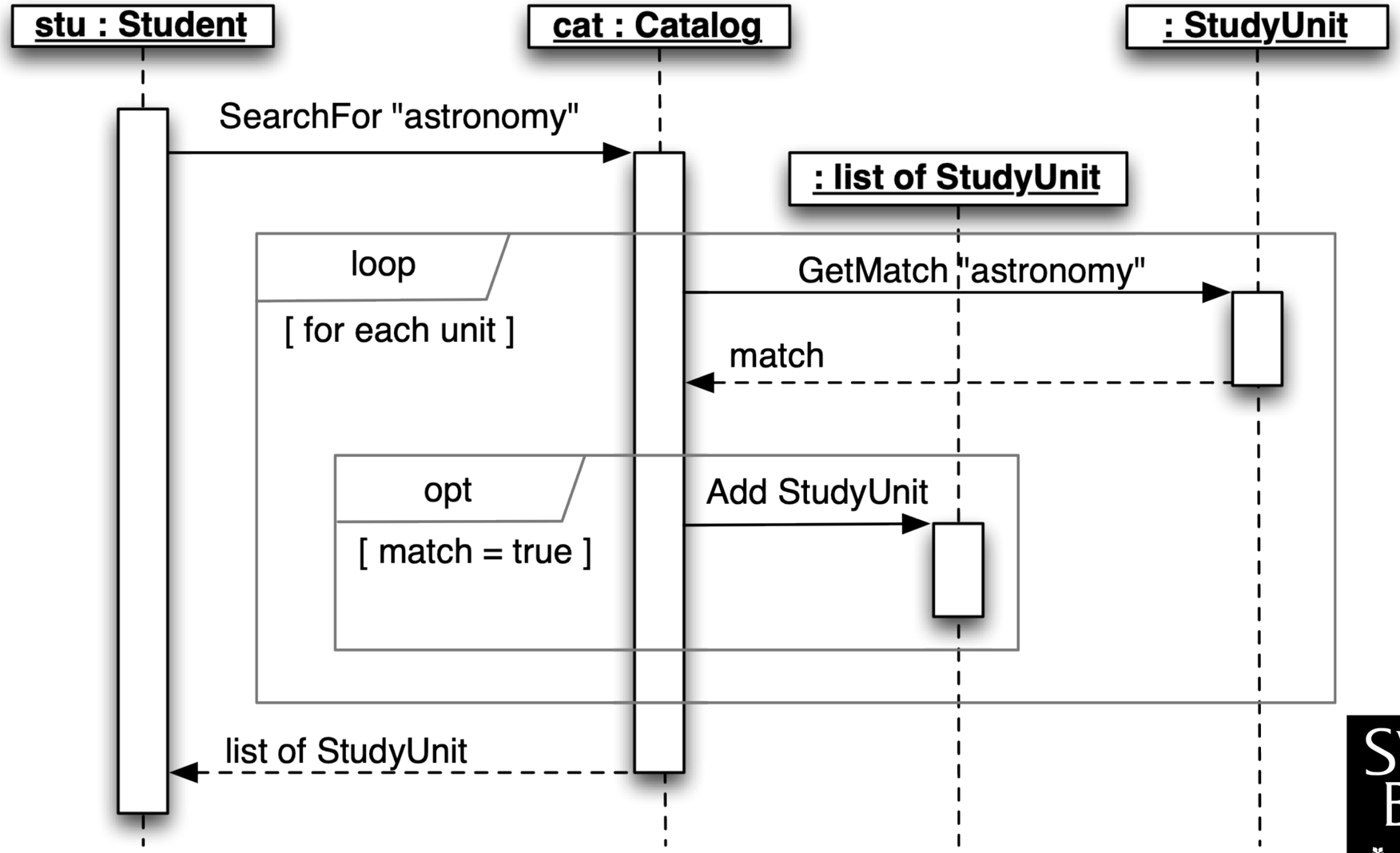


SWINBURNE UNIVERSITY OF TECHNOLOGY

Show alternatives to model if with else



Use loops to model repetition



SWINBURNE UNIVERSITY OF TECHNOLOGY

Take away message

- Responsibility driven design (RDD) focuses on object roles, responsibilities, and interactions

- There are many role stereotypes in RDD, watch out carefully

https://learn.microsoft.com/en-us/archive/msdn-magazine/2008/august/patterns-in-practice-object-role-stereotypes

- Effective designs ease the process of implementation, for teams and individual developers

- Communication is the key to consolidate understanding what you are building and why you are doing in that way

SWINBURNE UNIVERSITY OF TECHNOLOGY